**1 – introdução**

A comercialização de serviços através de meios eletrônicos tem crescido de forma exponencial nos últimos anos. De acordo com a pesquisa realizada pela PWC (PRICEWATERHOUSE COOPERS), uma das maiores prestadoras de serviços profissionais do mundo nas áreas de auditoria e consultoria, o principal fator responsável por esse crescimento é o preço, que tende a ser menor em lojas virtuais. Entretanto, ainda são encontradas certas dificuldades em alguns ramos de serviços, pois, dependendo de qual seja, deverá considerar inúmeras variáveis no momento de sua precificação.

Entre as diversas opções de serviços, a tradução de texto vem crescendo de forma gradativamente ao decorrer dos anos. Segundo pesquisa feita pela Common Sense Advisory(2017), líder de pesquisa e análise independente do setor de tradução, o mercado mundial de serviços de idiomas tem crescido a uma taxa anual de 5,52%, e um fator importante para este crescimento é a globalização mundial. Ao se tratar de assuntos comerciais, as relações lucrativas entre países com linguagens distintas cada vez se torna mais presentes e por isso exigem traduções cada vez mais precisas, que considerem não apenas a transposição de um idioma para outro, mas também a cultura, os aspectos políticos, econômicos e legais do país onde a comunicação será veiculada.

Diversas empresas do ramo de tradução possuem problemas ocultos no momento de calcular orçamentos para seus clientes. O termo “oculto” foi utilizado pois nem sempre está à percepção de todos, por isso até hoje o mercado é maioritariamente baseado na precificação definida por caracteres. Hoje, para a tradução de um arquivo de texto, as variáveis que são consideradas para o cálculo de um orçamento são em sua maioria o número de caracteres, laudas ou palavras que o documento possui, isso resulta na geração de um valor que avalia apenas números e não conteúdo, podendo retornar o mesmo preço para a tradução de um livro infantil e um documento judicial, sendo que claramente a complexidade no processo de tradução de cada um será divergente.

**2 – Objetivo Geral**

Propor e apresentar uma ferramenta seguindo uma nova metodologia para o cálculo de orçamentos de serviços de tradução de documentos de texto, visando o retorno de um valor compatível com o conteúdo presente em cada demanda e não apenas a quantidade de letras, palavras ou laudas. A proposta vem como uma melhoria pois se baseia apenas nas características e *features* que o documento apresenta, com isso o preço da tradução de um artigo científico fica compatível com sua complexidade, assim como uma revista em quadrinhos também ficará pois receberá a mesma análise.

A nova metodologia se torna uma forma mais rigorosa e exata, por isso pode vir a ser aplicada no real mercado do serviço de tradução. Deixando para trás a antiga metodologia que trazia algumas falhas em seu processo.

**2.1 – Objetivo Específico**

**2.1.1 – Pré-processamento**

Elaboração de um algoritmo utilizando a linguagem Python para o pré-processamento dos dados, esta análise lê um diretório de arquivos e retorna a quantidade de:

* Verbos;
* Substantivos;
* Páginas;
* Total de caracteres;
* Média de caracteres por página;
* Palavras;
* Palavras únicas;
* Palavras de um terceiro idioma;
* Caracteres numéricos;
* Índice de Facilidade de Leitura de Flesch.

de cada documento analisado. Após a análise, é criado um *Dataframe* contendo todos estes resultados e uma coluna informando o preço da tradução de cada um destes arquivos, esta coluna será essencial para a aprendizagem de máquina.

**2.1.2 – Processamento**

Foram utilizados os recursos de *Machine Learning* para a análise do *Dataframe* criado anteriormente, o algoritmo considera cada rótulo denominado de *feature* e “aprende” o padrão de precificação de acordo com cada um desses rótulos, ou seja, de qual forma cada um deles influencia diretamente no preço de um arquivo de acordo com a quantidade encontrada nele. Este padrão será aplicado pela Inteligência Artificial (IA) quando um novo arquivo nunca visto por ela for submetido, cada rótulo será analisado e o preço final será calculado de acordo com o que ela aprendeu em seu treinamento.

**3 – Referencial Teórico**

**3.1 – Python**

Para o desenvolvimento da aplicação, foi utilizada a linguagem de programação Python 3.7.0. A escolha desta linguagem foi feita devido ao fato de ser a que melhor atende à proposta do projeto por ser:

* Uma linguagem de alto nível muito acessível. Foi criada com a intenção de ser facilmente entendida por crianças (VAN RUSSON, 1999).
* Portadora de uma sintaxe excepcionalmente legível e com transparência sintática e semântica.
* Uma linguagem interpretada, o Python é cabível para exploração interativa. O programador pode experimentar e receber feedbacks imediatos.
* Portadora de um extenso acervo de bibliotecas gratuitas que oferecem inúmeras funções e métodos aplicáveis de acordo com a necessidade de cada projeto. Como por exemplo a biblioteca NLTK, que trabalha com Processamento de Linguagem Natural (PLN) e tem como objetivo a compreensão automática de linguagens humanas, de maneira que podem ser manipuladas por computadores.

De acordo com Mark Lutz (*Programming Python*, 2001), existem quatro motivos que resumem as razões para a linguagem Python ser utilizada, são eles:

1. **Qualidade**, pois ele torna fácil a maneira de codificar *softwares* que poderão ser reutilizados e mantidos;
2. **Produtividade**, pois a linguagem é otimizada para desenvolvimento ágil;
3. **Portabilidade**, pois a maioria dos programas em Python conseguem executar sem alterações em quase todos os computadores utilizados hoje em dia;
4. **Integração**, pois a linguagem foi projetada para ser integrada com outra ferramentas.

**3.2 – Natural Language Toolkit (NLTK)**

O NLTK é um conjunto de bibliotecas criada por Steven Bird, Edward Loper e Ewan Klein no ano de 2001 e tem como funcionalidade processamento textual, sua criação teve como base 4 (quatro) objetivos primários, são eles:

1. **Simplicidade**: Fornecer um framework intuitivo junto a blocos de construção substanciais, dotando os usuários de um conhecimento prático de PLN sem prender-se nas tediosas tarefas de "arrumação da casa" geralmente associadas com o processamento de dados linguísticos anotados.
2. **Consistência**: Fornecer um framework unificado com interfaces e estruturas de dados consistentes, e nomes de método facilmente conjecturáveis.
3. **Extensibilidade**: Fornecer uma estrutura que proporcione que novos módulos possam ser incluídos facilmente, incluindo implementações alternativas a diversas abordagens para uma mesma tarefa.
4. **Modularidade**: Fornecer componentes que possam ser utilizados sem a necessidade de compreender o restante do toolkit.

**3.2.1 – Tokenização**

Para melhor análise de informações contidas em documentos, existe a função *Tokenize* da biblioteca NLTK para efetuar a quebra de textos em palavras. Esta função cria um *array* com todas as palavras separadamente, sendo possível analisar e classificar uma por uma de acordo com sua categorização.

from nltk.tokenize import word\_tokenize   
tokenized\_word=word\_tokenize(new\_string)

Algoritmo 1 - Função Tokenize da biblioteca NLTK.

**3.4 – Categorização e classificação de palavras**

Para a classificação das palavras separadas no processo de Tokenização, existe a função *Pos\_tag* da biblioteca NLTK.

classificado=nltk.pos\_tag(tokenized\_word)

Algoritmo 2 - Função *Pos\_tag*

Esta função executa a chamada “*parts of speach tagging*”, ela também é conhecida por conter categorias lexicais pelo fato de analisar cada palavra e categorizar através de suas tags específicas que determinam e classificam cada palavra existente em um texto, são elas as classificações:

* CC: Conjunção Coordenativa
* CD: Dígito Cardinal
* DT: Determinador
* EX: Existencial (ex: ‘existe’, ‘há’)
* FW: Palavra Estrangeira
* IN: Preposição Conjunta/Conjunção Subordinada
* JJ: Adjetivo (ex: ‘grande’)
* JJR: Adjetivo Comparativo (ex: ‘maior’)
* JJS: Adjetivo Superlativo (ex: ‘o maior’)
* LS: Marcador de Lista (ex: ‘1)’ )
* MD: Modais (ex: ‘dever’, ‘poder’)
* NN: Substantivo Singular (ex: ‘mesa’)
* NNS: Substantivo Plural (ex: ‘mesas’)
* NNP: Substantivo Próprio Singular (ex: ‘Paulo’)
* NNPS: Substantivo Próprio Plural (ex: ‘Americanos’)
* PDT: Predeterminador (ex: ‘todas as crianças’)
* POS: Finais Possessivos
* PRP: Pronome Pessoal (ex: ‘eu’, ‘ele’, ‘ela’)
* PRP$: Pronome Possessivo (ex: ‘meu’, ‘dela’, ‘dele’
* RB: Advérbio (ex ‘silenciosamente’)
* RBR: Advérbio Comparativo (ex: ‘melhor’)
* RBS: Advérbio Superlativo (ex: ‘educadíssimo’)
* RP: Prático
* TO: Ações (‘ir à loja’)
* UH: Interjeições (ex: errrrrrrrm)
* VB: Verbo
* VBD: Verbo Pretérito
* VBG: Verbo, Gerúndio/Particípio Presente
* VBN: Verbo, Particípio Passado
* VBP Verbo, 1ª pessoa
* VBZ: Verbo, 3ª pessoa
* WDT: Determinador-WH (ex: ‘qual’)
* WP: Pronome-WH (ex: ‘quem’, ‘o que’)
* WP$: Pronome-WH Possessivo (ex: ‘de quem’)
* WRB: Advérbio-WH (ex: ‘onde’, ‘como’)

**3.5 – Rede Neural artificial (RNA)**

NELSON & ILLIGNWORTH (1990), definem as RNA’s como sendo técnicas inovadoras de processamento de informações, que têm como objetivo básico simular, em computadores, o funcionamento do sistema nervoso biológico.

Já Murillo Grübler (2018) apresenta uma definição mais técnica para as RNA’s como uma estrutura complexa interligada por elementos de processamento simples (neurônios), que possuem a capacidade de realizar operações como cálculos em paralelo, para processamento de dados e representação de conhecimento.

A Rede Neural Artificial é uma abstração da Rede Natural Biológica, ela serve de modelo para o aprendizado e a resolução de problemas complexos, tais como reconhecimento de fala e imagens, processamento de linguagem natural e previsão de valores.

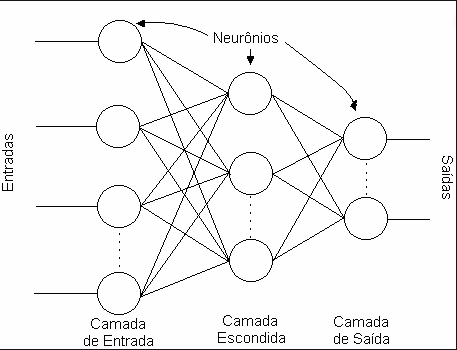


Figura 1 - Arquitetura *Perceptron*. Fonte – Fabio Mendes Soares

**3.5.1 – *Multi-Layer Perceptron* (MLP)**

O modelo de *Multi-Layer Perceptron* surgiu como uma evolução do modelo *Perceptron* criado em 1958 por Frank Rosenblatt (Van Der Malsburg, 1986), que é um dos modelos mais antigos modelos e lida com um único neurônio, classificando o resultado de forma linear. Já o MLP apresenta um modelo de *Perceptron* em múltiplas camadas. O grande diferencial que o MLP apresentada em relação ao *Perceptron* simples é a sua capacidade de solucionar problemas não linearmente separáveis, ou seja, que são necessárias mais de uma reta para que os grupos possam ser separados.

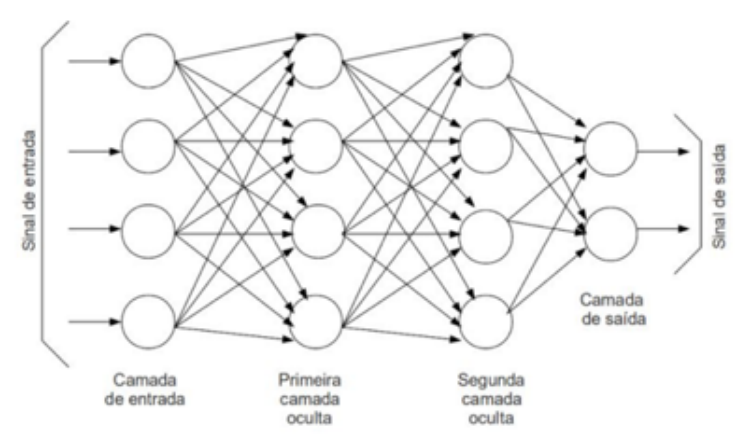
Este modelo possui uma arquitetura em camadas, que podem conter números variados de neurônios em cada uma. Elas são organizadas de forma que os neurônios de cada camada recebe como entrada a saída da camada anterior, o que é chamado de *feed-forward.*

Figura 2 - Arquitetura MLP (*Multi-layer Perceptron*). Fonte - Victor Gutemberg Oliveira Marques

**3.5.2 – Simple Linear regression**

A *Simple Linear Regression* é um método estatístico para obter uma fórmula com objetivo de prever valores de uma variável para outra onde há uma relação casual entre essas duas variáveis.

A fórmula para um gráfico linear se entende como  
 y = a + bx  
onde y é uma variável dependente e x é uma variável independente. Este modelo encontra a função linear de um problema e, com maior precisão possível, prevê o valor da variável dependente como função da variável independente.

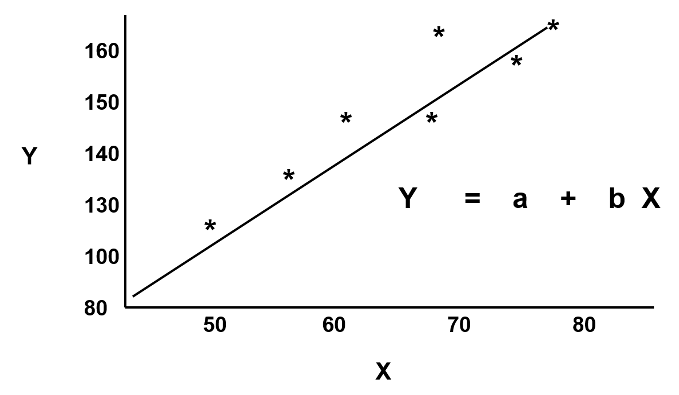


Figura 3 – *Simple Linear Regression*. Fonte - Boston University School of Public Health

**3.6 - Índice de Facilidade de Leitura Flesch**

A pontuação de um arquivo no Índice de facilidade de leitura Flesch determina o quão fácil é a leitura dele. O teste de legibilidade funciona contando a quantidade de palavras, sílabas e frases de um texto. Após fazer a contabilidade, é calculada a média de palavras por frases e a média de sílabas por palavras, seguindo a ideia de que palavras e frases menores são mais fáceis de serem lidas. Quanto mais alta for a pontuação, mais fácil é o texto para ser entendido.

A pontuação é encontrada através da fórmula:

  
Figura 4 - Fórmula do Índice Flesch. Fonte - Wikipédia, *Flesch Reading Ease*

ou seja, 206,835 menos 1,015 vezes a divisão do número de palavras pelo total de frases, menos 84,6 vezes a divisão do total de sílabas pelo total de palavras.

Após ser encontrado o resultado, ele é consultado na tabela de pontuação para encontrar em qual nível de legibilidade o texto se adequa.

|  |  |
| --- | --- |
| **Escala Flesch** | **Nível de legibilidade** |
| 0-29 | Muito Difícil |
| 30-49 | Difícil |
| 50-59 | Razoavelmente Difícil |
| 60-69 | Padrão |
| 70-79 | Razoavelmente Fácil |
| 80-89 | Fácil |
| 90-100 | Muito Fácil |

Tabela 1 - Escore Flesch distribuído de acordo com nível de escolaridade. Fonte - Lucas Lobato.

.

**4 – Trabalhos Relacionados**

**4.1 – Mineração de Dados**

No estudo feito por Cássio Oliveira Camilo e João Carlos da Silva, estudantes do Instituto de Informática Universidade Federal de Goiás, foi apresentado de forma aprofundada os conceitos fundamentais da Mineração de Dados. A mineração de dados é uma das tecnologias mais promissoras da atualidade pelo fato de dezenas, e às vezes centenas de milhões de reais serem gastos pelas companhias na coleta dos dados e por fim nenhuma informação útil é identificada.

No estudo é trazido as principais áreas nas quais a Mineração de Dados é aplicada de forma satisfatória como por exemplo na Retenção de Clientes, Bancos, Cobranças, Telemarketing, Eleitoral, Medicina, Segurança, RH, Agência de Viagens e muito mais.

A mineração de Dados tornou-se uma ferramenta de apoio com papel fundamental na gestão da informação dentro das organizações. A maneira tradicional de manipulação de dados e análise de informações tornou-se inviável devido ao grande volume de dados, por isso, descobrir padrões implícitos e relacionamentos em repositórios que contêm um grande volume de dados de forma manual, deixou de ser uma opção.

Não há dúvidas de que é uma área extremamente promissora e que, apesar dos resultados já obtidos, ainda tem muito a oferecer.

**4.2 – Importância do Serviço de Tradução**

O estudo realizado por Cláudio Luiz Oliveira graduado em Letras - Português/Espanhol e Respectivas Literaturas pela Universidade Federal do Acre enfatiza a importância dos serviços de tradução feita por um profissional, levando em consideração que a tradução não se trata de uma mera reprodução de um texto em outra língua, o tradutor assume um papel de destaque em meio a esse contexto tendo que, além de transpor as palavras em outra língua, também organizar o texto de forma que fique com o mesmo sentido que o original.

Segundo Basnett (2003, p. 54) “a tradução não é a substituição de elementos lexicais e gramaticais entre línguas”, mas sim uma forma de tornar um texto legível a um leitor que desconhece a língua do texto original.

O estudo reforça a complexidade de uma tradução e que a contratação de serviços de especialistas para realizá-la é primordial para que o texto mantenha sua essência.

**5 – Metodologia**

**5.1 – Linguagem e ferramentas**

A aplicação foi desenvolvida utilizando a linguagem Python na versão 3.7.0 e aplicando essencialmente a biblioteca NLTK para o processamento de texto, classificação, tokenização, análise e raciocínio semântico. Após a tokenização, foi aplicada a função *pos\_tag* exclusivamente para a classificação de cada palavra presente nos textos que foram analisados.

A IDE (Integrated Development Environment) selecionada para o desenvolvimento do projeto foi a Visual Studio Code na versão 16.3.4. Sua seleção foi devido ao fato de ser um editor de código-fonte desenvolvido pela Microsoft que inclui suporte para depuração, controle Git incorporado, realce de sintaxe, complementação inteligente de código, *snippets* e refatoração de código. É um software livre e de código aberto.

**5.2 – Geração do Dataframe**

Para o treinamento da RNA, inicialmente foi feito o algoritmo para a geração do *Dataframe* contendo todas as *features* de cada arquivo de um diretório separado apenas para este fim, essas *features* são as características textuais dos documentos.

Este *Dataframe* é composto por 12, uma coluna por *feature*, elas são as informações essenciais para o treinamento da Inteligência Artificial (IA). São elas:

* Nome do Arquivo;
* Média de caracteres por páginas;
* Caracteres;
* Número de páginas;
* Número de palavras;
* Dígitos cardinais;
* Palavras estrangeiras;
* Palavras únicas;
* Substantivos;
* Verbos;
* Pontuação no Índice Flesch;
* Preço do arquivo.

A utilização da biblioteca NLTK tornou prático o processo de contabilização de cada *feature*. Dentro de uma estrutura de repetição, sempre quando era encontrada uma palavra que foi classificada como “Verbo”, por exemplo, era acrescido 1 (um) no contador de verbos, e assim foi feito para cada *feature*.

No caso do número de palavras, número de páginas, caracteres e média de caracteres por páginas, a análise foi feita sem o uso da NLTK. Para ser possível analisar cada uma destas informações, foi extraído o texto de cada arquivo e a análise foi feita sob ele.

O Índice Flesch foi encontrado através da fórmula citada no tópico 3.6, o cálculo foi feito com o auxílio da biblioteca Textstat utilizanddo a função *flesch\_reading\_ease,*  que possui como retorno a pontuação do texto extraído do arquivo.

Ao fim de cada iteração e da contabilização de cada rótulo, os resultados foram inseridos em um Dicionário que reteve as informações de cada arquivo analisado. Após o fim das iterações, ou seja, o algoritmo já fez a análise de todos os arquivos do diretório, o Dicionário com as informações de todos os documentos foi inserido no *Dataframe*.

**5.3 – Precificação dos arquivos**

Para ser preenchida a coluna de preços dos arquivos, foi necessário criar uma nova forma de precificar todos estes documentos para o treinamento da IA, levando em consideração a nova proposta de metodologia que tem como seu principal foco a complexidade da tradução do texto.

Para isso, foi utilizada a pontuação de cada um no Índice Flesch e a categoria em que eles se encaixaram.

Foi atribuído um peso a cada feature e em cada classificação da tabela do Índice Flesch para atuarem diretamente no preço final. A equação aplicada foi:

Preço = ((Quantidade de caracteres \* 0,005) + (Quantidade de páginas \* 5) + (Quantidade de palavras \* 0,002) + (Quantidade de dígitos cardinais \* 0,00005) + (Quantidade de palavras estrangeiras \* 0,002) + (Quantidade de palavras únicas \* 0,0003) + (Quantidade de substantivos \* 0,007) + (Quantidade de verbos \* 0,002)) \* fator de multiplicação do Índice Flesch.

Pelo fato de a tabela de pontuação considera o menor valor como mais difícil e o maior valor como o mais fácil, cada categoria recebeu um fator de multiplicação que aumentava de acordo com sua complexidade. Na tabela abaixo é indicada cada atribuição:

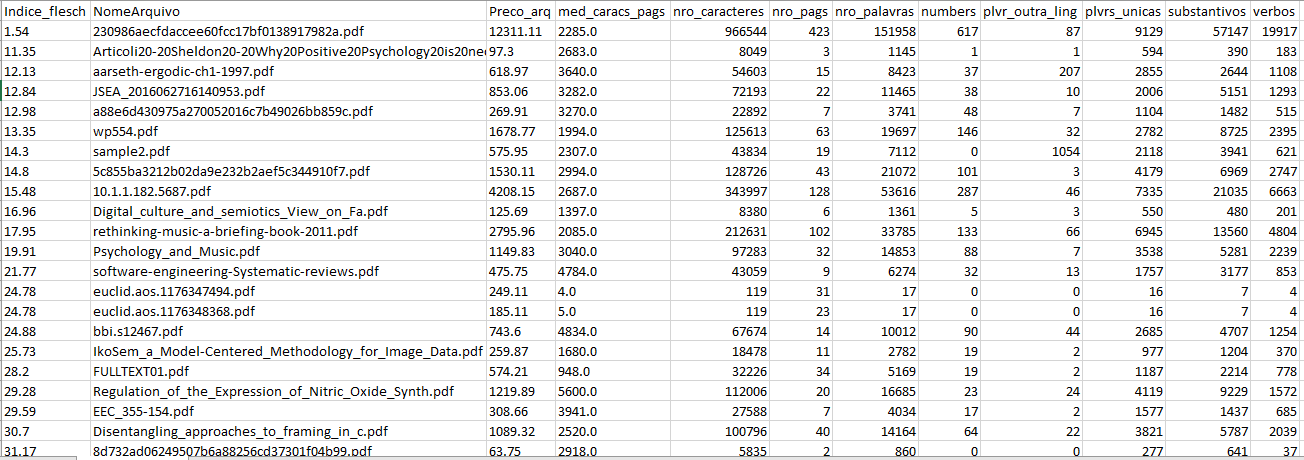
|  |  |
| --- | --- |
| **Nível de legibilidade** | **Fator Multiplicador** |
| Muito Difícil | 1.6 |
| Difícil | 1.4 |
| Razoavelmente Difícil | 1.2 |
| Padrão | 1.0 |
| Razoavelmente Fácil | 0.8 |
| Fácil | 0.6 |
| Muito Fácil | 0.4 |

Tabela 2 - Fator de Multiplicação para cálculo do preço

Este fator foi o principal diferencial para definir o preço final de cada arquivo, focando principalmente na complexidade mas considerando também as quantidades de *features* de cada um, algo que influencia diretamente no tempo de manipulação deles.

**5.3 – Treinamento da Inteligência Artificial**

Com o *Dataframe* completo criado, ele foi utilizado nos algoritmos de RNA para seu treinamento. Este algoritmo para a criação do *Dataframe* foi executado analisando um diretório que continha 150 arquivos, destes 150 arquivos, 30% foi utilizado para teste e 70% foi utilizado para treinamento da IA..

  
Figura 5 - Dataframe final

Ao analisar a proposta do projeto, foi optado pelo uso dos algoritmos de RNA *Multi-Layer Perceptron* e *Simple Linear Regression.* Ambos foram executados sob o *Dataset* com as configurações para serem analisados e ser definido qual seria utilizado. A decisão se fez através dos gráficos que foram gerados informando a performance de cada um.

A aplicação dos algoritmos de treinamento de IA foram utilizados através da biblioteca *sklearn,* para a MLP foi aplicada a classe *MLPRegressor()* e para a Simple Linear Regression foi utilizada a classe *LinearRegression().*

from sklearn.linear\_model import LinearRegression

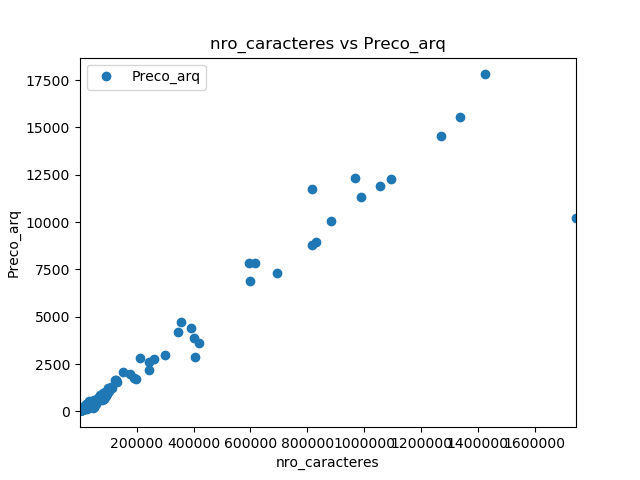
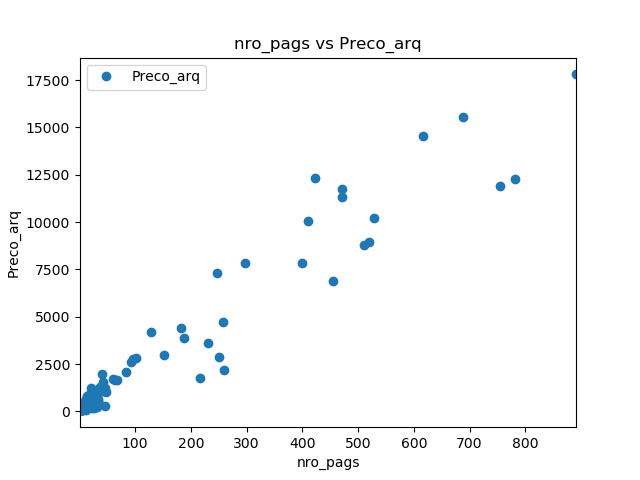
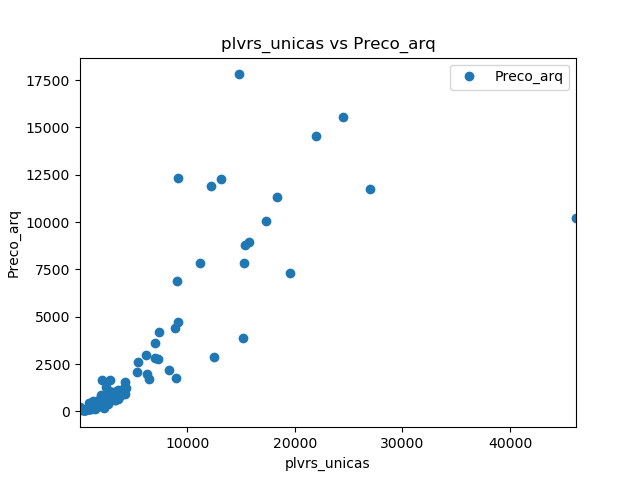
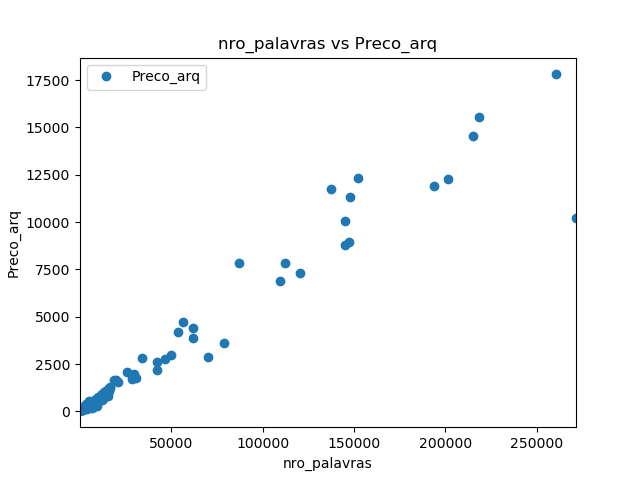
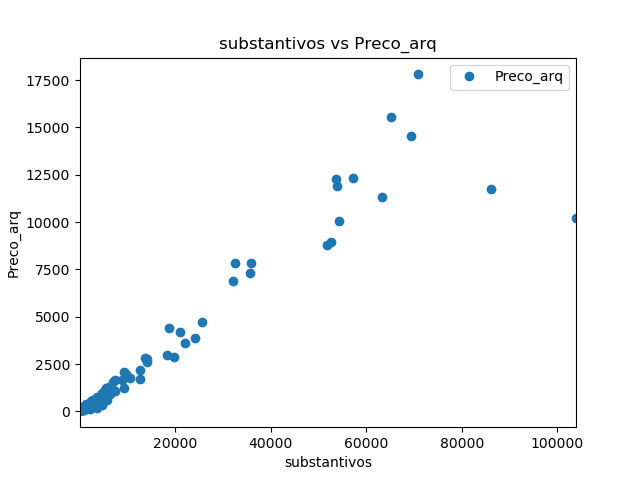
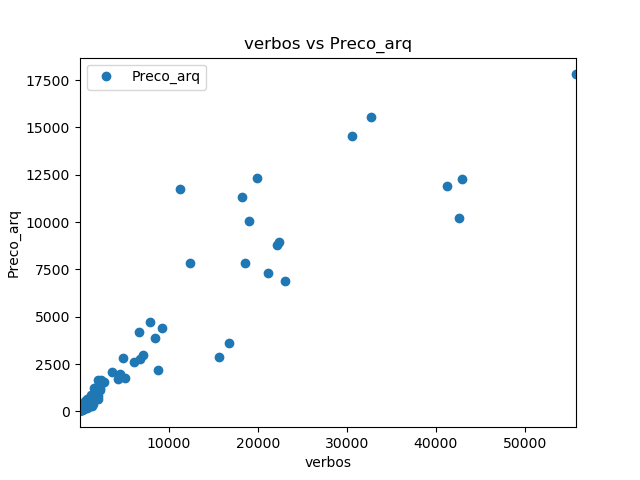
from sklearn.neural\_network import MLPRegressor

Algoritmo 3 - Import da classe LinearRegression e MLPRegressor da biblioteca sklearn

**6 – Resultados Encontrados**

**6.1 – Análise de Dados**

Para melhor análise dos dados e melhor entendimento da forma em que cada *feature* implica no preço, foi utilizada a função *pyplot* da biblioteca *Matplotlib* para serem gerados gráficos ilustrando cada resultado.

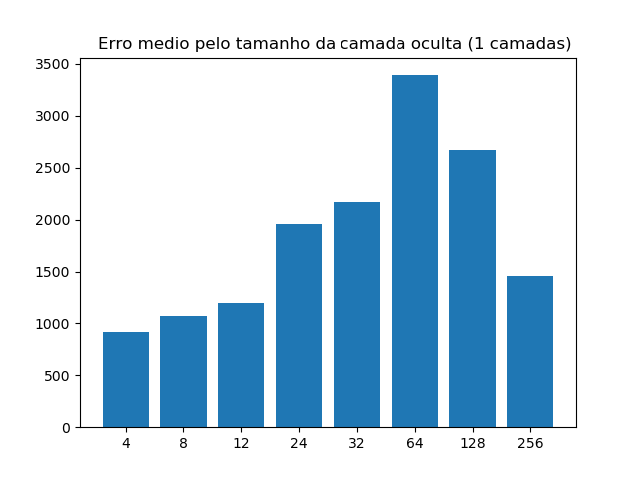
*  
Figura 5 – Gráfico ilustrando influência do Número de Caracteres no Preço do arquivo  
Figura 6 – Gráfico ilustrando influência do Número de Páginas no Preço do arquivo   
Figura 7 – Gráfico ilustrando influência do Número de Palavras Únicas no Preço do arquivo   
Figura 8 – Gráfico ilustrando influência do Número de Palavras no Preço do arquivo   
Figura 9 – Gráfico ilustrando influência do Número de Substantivos no Preço do arquivo   
Figura 10 – Gráfico ilustrando influência do Número de Verbos no Preço do arquivo*

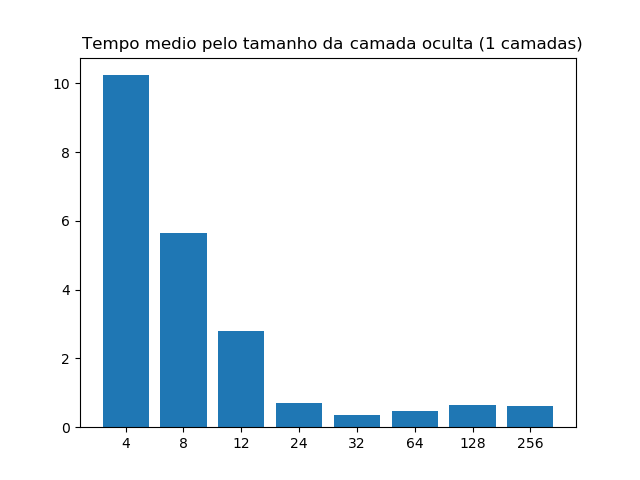
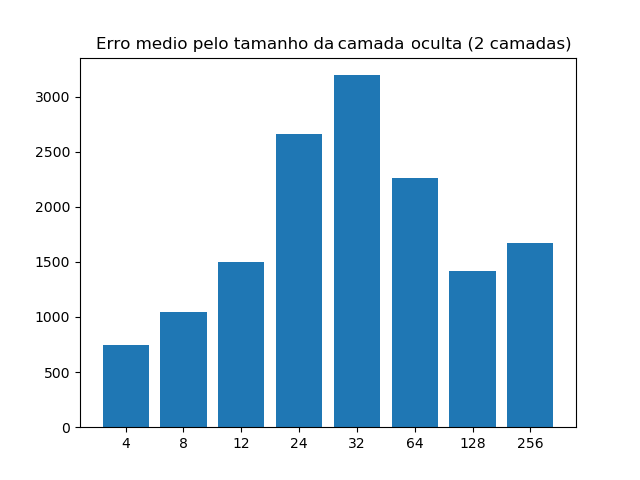
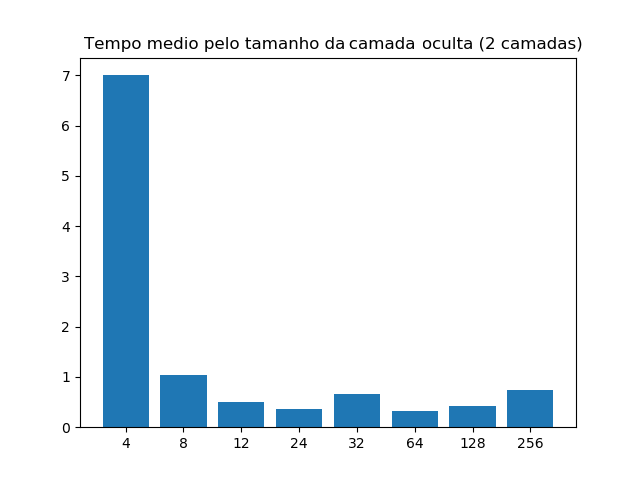
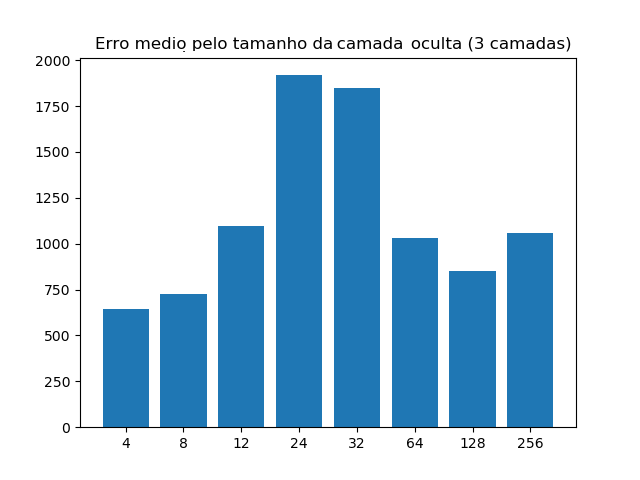
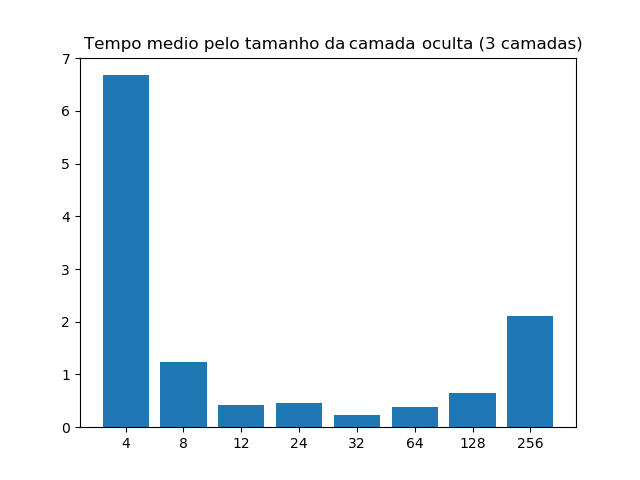
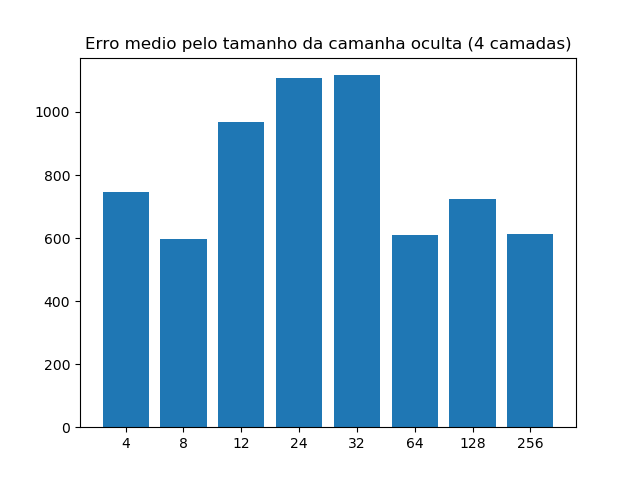
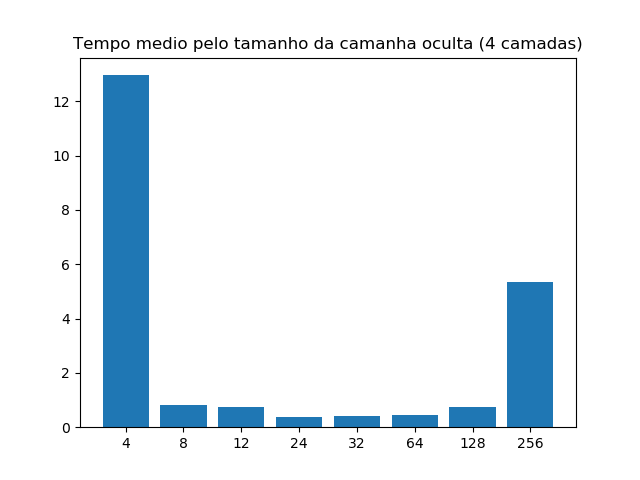
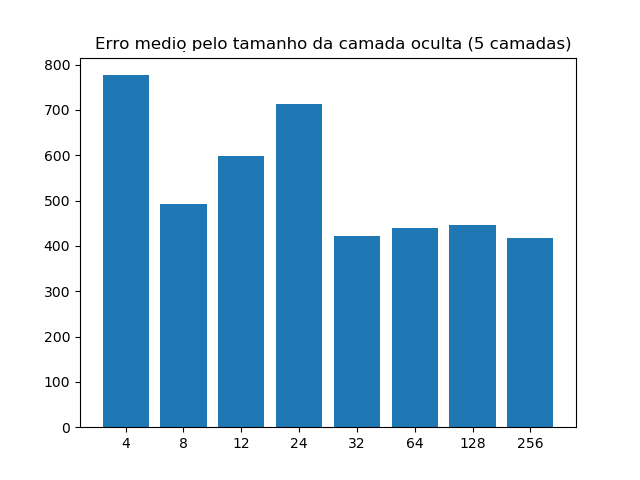
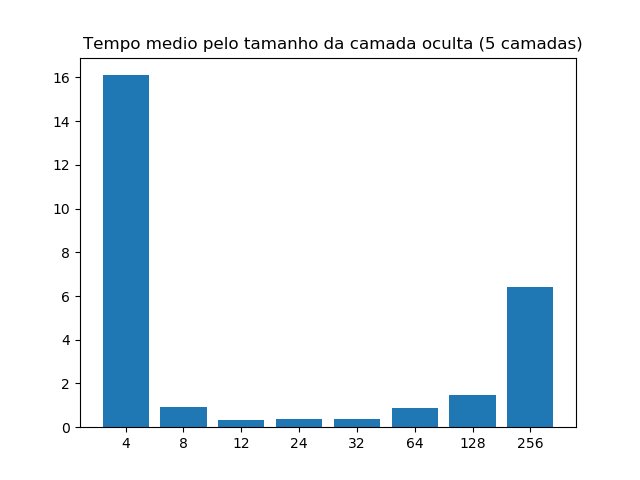
**6.2 – Aplicação MLP**

Para o treinamento da Rede Neural Artificial, foi aplicado o algoritmo MLP no *Dataframe* com os dados dos arquivos.

A calibragem do algoritmo foi definida através de testes com cada camada e a quantidade de neurônios em cada uma. Utilizando ainda a função *pyplot* da biblioteca *Matplotlib* para ilustrar os resultados de cada teste.

Os testes foram feitos executando o algoritmo MLP 50 (cinquenta) vezes em uma estrutura de repetição, cada resultado foi registrado para ser feita a média dos resultados ao final. Foram feitos testes com 1 (uma), 2 (duas), 3 (três), 4 (quatro) e 5 (cinco) camadas com a quantidade de 4 (quatro), 8 (oito), 12 (doze), 24 (vinte e quatro), 32 (trinta e dois), 64 (sessenta e quatro), 128 (cento e vinte e oito) e 256 (duzentos e cinquenta e seis) neurônios em cada camada a cada iteração. Os resultados encontrados por camada foram:

  
*Figura 11 – Gráfico ilustrando Erro Médio dos resultados do algoritmo testado com Uma camada de acordo com a quantidade de neurônios configurados nela.*

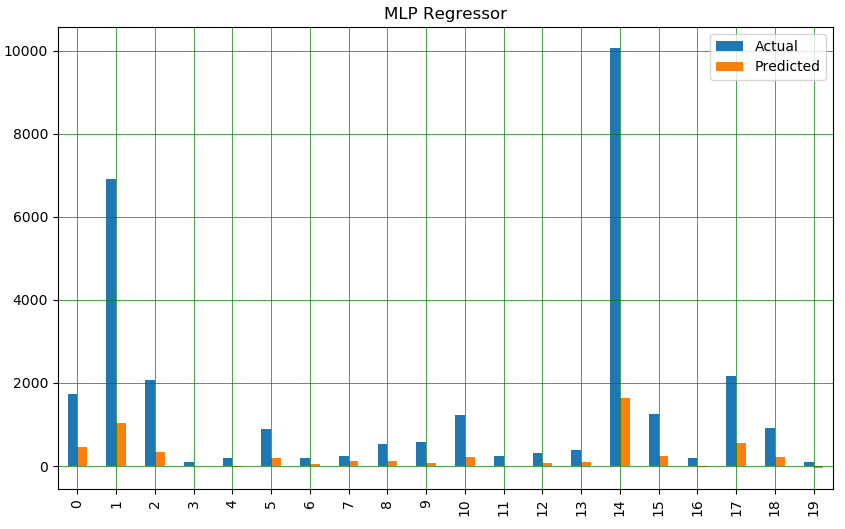
  
*Figura 12 – Gráfico ilustrando Tempo Médio dos resultados do algoritmo testado com Uma camada de acordo com a quantidade de neurônios configurados nela.* *Figura 13 – Gráfico ilustrando Erro Médio dos resultados do algoritmo testado com Duas camadas de acordo com a quantidade de neurônios configurados nelas.* *Figura 13 – Gráfico ilustrando Tempo Médio dos resultados do algoritmo testado com Duas camadas de acordo com a quantidade de neurônios configurados nelas.* *Figura 14 – Gráfico ilustrando Erro Médio dos resultados do algoritmo testado com Três camadas de acordo com a quantidade de neurônios configurados nelas.* *Figura 15 – Gráfico ilustrando Tempo Médio dos resultados do algoritmo testado com Três camadas de acordo com a quantidade de neurônios configurados nelas.* *Figura 16 – Gráfico ilustrando Erro Médio dos resultados do algoritmo testado com Quatro camadas de acordo com a quantidade de neurônios configurados nelas.* *Figura 17 – Gráfico ilustrando Tempo Médio dos resultados do algoritmo testado com Quatro camadas de acordo com a quantidade de neurônios configurados nelas.* *Figura 18 – Gráfico ilustrando Erro Médio dos resultados do algoritmo testado com Cinco camadas de acordo com a quantidade de neurônios configurados nelas.* *Figura 19 – Gráfico ilustrando Tempo Médio dos resultados do algoritmo testado com Cinco camadas de acordo com a quantidade de neurônios configurados nelas.*

Observando os resultados de todos os gráficos e comparando os valores por Tempo e Erros Médios por quantidade de neurônios, foi possível concluir que a melhor calibração para a MLP foi de 4 (quatro) camadas contendo 64 (sessenta e quatro) neurônios em cada. Tendo em vista que a comparação com a Média de Erros e com o Tempo de processamento de 3 (três) e 5 (cinco) camadas, a performance da calibragem de 4 (quatro) camadas apresentou o melhor resultado em ambos os casos, apresentando menos erros que 3 (três) e 5 (cinco) e melhor tempo que ambas.

regressor\_MLP=MLPRegressor(*hidden\_layer\_sizes*=(64,64,64,64), *max\_iter*=10000)

Algoritmo 4 – Aplicação da classe MLPRegressor() da biblioteca sklearn

Após determinar a calibragem do algoritmo, foi possível plotar os resultados encontrados pelo MLP mostrando os Preços reais dos arquivos (Actual - Azul) e os preços previstos para eles pela MLP (Predicted - Laranja).

  
*Figura 20 – Gráfico ilustrando a comparação entre os Preços Previstos pelo MLP e os Reais.*

**6.3 – Aplicação Simple Linear Regression**

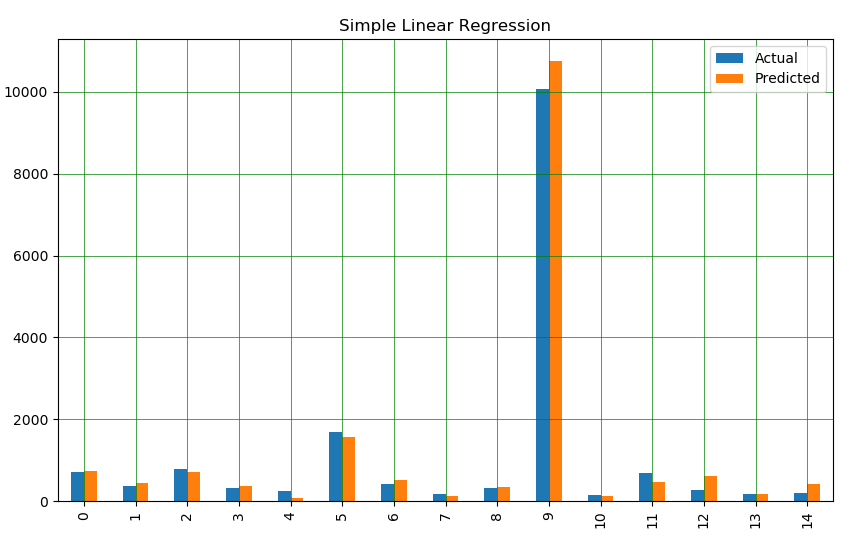
Para a tentativa de melhoria de resultados das previsões da IA, foi aplicado mais um algoritmo no Dataframe com os dados dos arquivos, o *Simple Linear Regression.*

regressor = LinearRegression()

Algoritmo 5 – Aplicação da classe LinearRegression() da biblioteca sklearn

O algoritmo não necessita calibragem igual o MLP pelo fato de ele analisar a relação entre as variáveis presentes no problema.

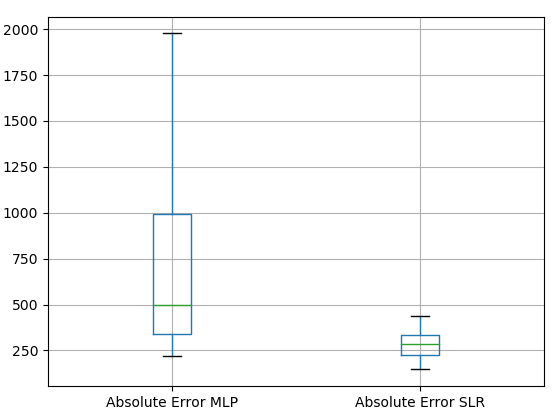
Após a execução do algoritmo, foi possível visualizar os resultados retornados por ele utilizando a função *pyplot*, ilustrando os valores reais dos arquivos analisados (Actual – Azul) e os valores previstos pelo algoritmo (Predicted – Laranja).

  
*Figura 21 – Gráfico ilustrando a comparação entre os Preços Previstos pelo Simple Linear Regression e os Reais.*

**6.4 – MLP vs Simple Linear Regression**

Observando os resultados retornados de cada algoritmo, é possível verificar uma porcentagem de assertividade maior do Simple Linear Regression comparando com o MLP.

Foi possível verificar de forma mais clara utilizando o gráfico comparativo de erros gerado pela função *boxplot* da biblioteca *Matplotlib.*

  
*Figura 22 – Gráfico ilustrando a comparação entre os Erros de previsão de preços do MLP e do SLR.*

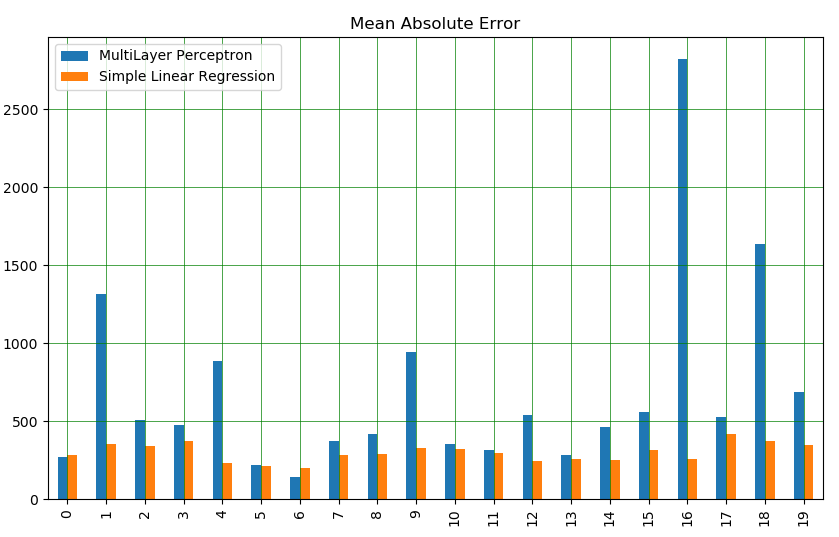
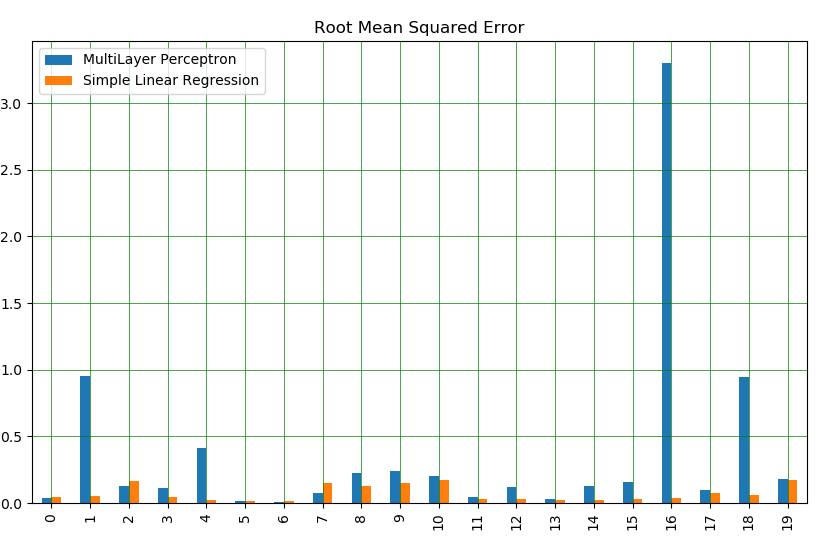
O gráfico ilustra a distribuição de dados e o comparativos entre os dois conjuntos de resultados, aplicando no problema do projeto ele apresenta a variância entre os erros dos valores previstos pelos algoritmos MLP e *Simple Linear Regression.*

Quanto maior é a espessura entre o primeiro quartil e o terceiro quartil, maior é a variância de erros. Pode-se observar que o primeiro e o terceiro quartil possuem uma longa distância no gráfico referente ao MLP, já no que se refere ao SLR a distância é menor, ou seja, apresentou menos erros.

**7 – Conclusão**

Com todos os testes feitos e resultados analisados, foi visto que o melhor retorno de valores foi provido pelo algoritmo *Simple Linear Regression*, devido ao fato de as variáveis presentes no problema apresentarem influência linear em relação ao preço, como é possível verificar nos gráficos das imagens 5 (cinco) à 10 (dez), cada *feature* influencia no preço de forma linear e crescente, ou seja, quanto maior for a quantidade daquela *feature*, maior será o valor da tradução.

Comparando visualmente os resultados de cada algoritmo, é possível enxergar de forma mais clara a diferença entre os erros do MLP e do SLR. Para melhor visualização, foram plotados dois gráficos utilizando métricas de *mean absolut error* e *mean squared error* que apresentam a medida da diferença entre duas variáveis contínuas e a média quadrática dos erros entre dois valores estimados, respectivamente.

*Figura 23 – Gráfico ilustrando a comparação entre os Erros dos algoritmos MLP e SLR aplicando o Mean Absolute Error.* *Figura 24 – Gráfico ilustrando a comparação entre os Erros dos algoritmos MLP e SLR aplicando o Mean Square Error.*

**8 – Referências Bibliográficas**

RODRIGUEZ, M.; PETERSON, R. M.; KRISHNAN, V. Social Media's Influence on Business-To-Business Sales Performance. Journal of Personal Selling & Sales Management, v. 32, n. 2, p. 365-378, Summer 2012 2012. ISSN 0885–3134 (print) / ISSN 1557–7813 (online). Disponível em: <http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=78110794&site=ehostlive >. Acesso em: 09 abr. 2019.

PWC (PRICEWATERHOUSECOOPERS) (2016), “Total Retail Brasil 2016”. Disponível em https://www.pwc.com.br/pt/setores-de- atividade/varejo-e-consumo/assets/2016/total\_retail\_16\_brasil.pdf>. Acesso em 09 abr. 2019.

DINO DIVULGADOR DE NOTÍCIAS (2017), “Com tecnologias cada vez mais avançadas, o mercado de serviços de tradução não para de crescer”. Disponível em <https://exame.abril.com.br/negocios/dino/com-tecnologias-cada-vez-mais-avancadas-o-mercado-de-servicos-de-traducao-nao-para-de-crescer-shtml/>. Acesso em 15 out. 2019.

Van Rossum, 1999 van Rossum, G. (1999). Computer programming for everybody. Technical report, Corporation for National Research Initiatives. <http://www.python.org/doc/essays/ cp4e.html> Acesso em 19 out. 2019.

CÁSSIO OLIVEIRA, C.; CARLOS DA SILVA, J.; Mineração de Dados: Conceitos, Tarefas, Métodos e Ferramentas. Agosto, 2019. Disponível em: <https://rozero.webcindario.com/disciplinas/fbmg/dm/RT-INF\_001-09.pdf> Acesso em 20 out. 2019

VICTOR GUTEMBERG OLIVEIRA MARQUES.(2017) “AVALIAÇÃO DO DESEMPENHO DAS REDES NEURAIS CONVOLUCIONAIS NA DETECÇÃO DE OVOS DE ESQUISTOSSOMOSE”. Disponível em <https://www.cin.ufpe.br/~tg/2017-1/vgom\_tg.pdf>. Acesso em 15 nov. 2019.

LUCAS LOBATO, BEATRIZ SANTANA CAÇADOR , MARIA FLÁVIA GAZZINELLI .(2013) “LEGIBILIDADE DOS TERMOS DE CONSENTIMENTO LIVRE E ESCLARECIDO EM ENSAIOS CLÍNICOS”. Disponível em <http://www.scielo.br/pdf/bioet/v21n3/a20v21n3.pdf>. Acesso em 15 nov. 2019.

Lutz, Mark. *Programming python*. " O'Reilly Media, Inc.", 2001.

NELSON, Marilyn M. & ILLINGWORTH, W. T. A Praticai Guide to Neural Nets. Addison-Wesley Publishing Company, Inc. USA, 1991.

Grübler, Murillo. “Entendendo o funcionamento de uma Rede Neural Artificial”. Disponível em: <https://medium.com/brasil-ai/entendendo-o-funcionamento-de-uma-rede-neural-artificial-4463fcf44dd0>

Van Der Malsburg, C. "Frank Rosenblatt: principles of neurodynamics: perceptrons and the theory of brain mechanisms." *Brain theory*. Springer, Berlin, Heidelberg, 1986. 245-248.

Simple linear regression. Disponível em: <http://www.statstutor.ac.uk/resources/uploaded/slregression.pdf>

BASSNETT, S. Estudos de tradução: Fundamentos de uma disciplina. Tradução de Vivina de Campos Figueiredo. Lisboa: Fundação Calouste, 2003. Acesso em 29 nov. 2019.

Claudio Luiz Oliveira. “A IMPORTÂNCIA DA TRADUÇÃO:REFLEXÕES SOBRE O PAPEL DO TRADUTOR”.(2017).Disponível em: <https://periodicos.ufac.br/index.php/COMMUNITAS/article/download/1109/pdf/> Acesso em 29 nov. 2019.